Harnessing the power of XML to Achieve a Successful Data Migration

Tony Holtham

Within any reasonable sized organisation, an established enterprise application will manage such a complexity of data and relationships between the data, that when a new application is deemed necessary, most businesses have little confidence that data migration will be a total success. This paper examines how XML can be used to deliver a new approach that increases the success of complex data migration projects and allows the old system to be retired sooner.

The complexity of data migration means that a big-bang approach to data migration is not usually seriously considered. This forces a project-by-project approach to be adopted which results in the existing system being active for a considerable amount of time. Even when the migration is deemed as complete, the system is often kept alive and maintained 'just in case' some of the data was omitted or mishandled. To address these issues, we must move away from the usual proprietary approach to data migration and look to applying proven technology in a number of discrete and manageable steps.

Manageable Steps

The first step is to make sure that all data held in the source system is captured into a neutral, non-propriety format. Over the past decade XML has emerged as an ideal format for ensuring longevity of access to data coupled with the ability to use a range of commercial tools.

With all the data stored in a neutral format, the subsequent steps can then be defined in complete confidence that all the data is captured and ready for migration. Should any of this data not be migrated to the new system, it will always be available for future use, without requiring access to the source application.

What Is XML?

XML is the (e**X**tensible **M**arkup **L**anguage). XML is a documented standard defined by the World Wide Web Consortium (W3C) and is now in wide use across many industries for facilitating the interchange of data between computer applications. XML is actually a meta-language that allows the creation of a standardised set of rules for adding structure to any form of data using a system of markup tags (elements). Any organisation can create their own markup vocabulary (called an XML Schema or XML Document Type Definition - DTD), and XML ensures that the structure will be intelligible to anyone else who consults the XML Schema/DTD document. More importantly, by referring to an XML Schema/DTD, any software that is XML-aware is able to automatically manipulate the data without needing advance knowledge of the structure.

XML is extensively used to meet the requirements of organisations for industry-specific markup, vendor-neutral data exchange, media-independent publishing, one-on-one marketing, workflow management in collaborative authoring environments, and the processing of documents by intelligent clients. XML is fully internationalised for both European and Asian languages, with all conforming processors required to support the Unicode character set in both its UTF-8 and UTF-16 encodings. The language is designed for the quickest possible client-side processing consistent with its primary purpose as an electronic publishing and data interchange format.

XML instances are made up of storage units called *entities*, which contain either parsed or unparsed data. Unparsed data consists of images, audio, video etc whilst parsed data is made up of *characters*, some of which form the *character data* in the document, and some of which form *markup*. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on that storage layout and structure. The very nature of XML is that it is a structured document format, in that it represents not only the information to be exchanged, but the metadata encapsulating its meaning and the structure of the information to be exchanged.

Why Use XML for Data Migration?

Separates formatting and processing from data

XML represents both information and the metadata about that information. It does not specify any particular manner for how the data should be processed or provide any constraints for mechanisms with which to handle the information. XML documents simply encode information and their metadata without specifying how the information is to be processed or displayed. This singular feature is a tremendous benefit for those looking for a cross-platform and cross-device language that is meant for simply encoding data and its structure.

This capability of XML to separate process and data content results in what is often termed as “future-proof” or “loosely coupled”. Future proofing means that no future changes in the data exchange layer should affect the programming layer, and vice-versa. Loosely coupled systems allow for “arms-length” exchange of information where one party does not need to know details of how the other party plans to process the information. This allows for changes in any of the presentation, process, or data layers without affecting the other layers.

XML instances can be validated

XML instances can be validated for correctness and come with error and validity checking built-in. The DTD or schema that is referred to by an XML document can guarantee, at time of document creation, that all the elements are correctly specified, in the correct order. Instances can be validated at time of creation or at time of receipt and be rejected or accepted on an automated basis without human intervention. At design-time, errors can be fixed before transmission, and on receipt, errors can be sent back to the sender for further human processing, with an exact pinpoint as to where the error has occurred.

Validity checking also comes at a very low cost, if not free, since most parsers on the market are available as open source and come with validation built-in.

Data can be viewed with simple tools, such as browsers

XML can be visualised using a number of low cost tools or methods:

- Internet browsers - native viewing of XML or by using XSL or CSS to render on the browser
- Conversion of XML to HTML – performed at server-side, run-time or in batch using XSL or other methods
- Use of specialised Java applications - to render XML in a browser

Whilst it is possible to use freely available browsers to visualise XML, more important are the business applications that understand XML and can interpret the documents in a manner that is relevant to the context in which they were produced. After all, if a business user is entering a purchase order in a system, they will be unable to debug an XML document if the other party rejects it. Instead, the tool used to create the document will have to interpret the results and present them back to the user in a consumable manner. Although such functionality is rare within the current generation of applications, it is starting to emerge in new applications and will continue to increase as XML becomes ever more pervasive.

License-free, platform-neutral, software independent, and widely supported

XML is a technology that has no single owner or point of commercial licensing. As such, it can be freely implemented in any application or usage scenario that an organisation sees fit, without incurring licensing costs. Due to the separation of process from content, it is also a good example of a platform-neutral data format.

In addition, XML is widely supported by all manner of individuals and organisations. As a truly open-source and open-process technology, XML provides implementers a wide base of resources that can provide assistance. Rather than being constricted to getting technical support and assistance from a single company, XML provides implementers the opportunity to obtain an XML-based product from

one company, implementation services from a second company, and support and ongoing maintenance from yet another company. This is the essence of the open-source movement.

The processing technology is widespread and easily available

Since XML is a structured document that shares many of the processing and parsing requirements of SGML and HTML, there are plenty of available parsers. Many of these parsers are now an integral part of generally available browsers and server-side agents

However, while XML processing tools are becoming inexpensive and ubiquitous, the implementation of these tools is not a free or painless process. Processing XML documents does not stop at simply parsing an XML file. The data from those documents needs to be acted upon. For most applications, parsing an XML document is just the first step of many. However, standard tools at least remove from programmers the worry of parsing the document.

Flexible

Extensibility is the ability to define specific vocabularies and metadata. Rather than being fixed in describing a particular set of data, XML in conjunction with its DTDs and Schema is able to define any of a number of documents that together form a language of its own.

Adding additional elements and attributes can easily extend XML instances. While XML is fairly simple in nature (it only needs to follow basic syntax rules to be considered “well-formed”), one of the biggest features of the language is its ability to provide a means for guaranteeing the validity of a document.

Human readable

Humans can easily read XML, and well-designed XML should be developed with readability in mind. Human readability not only makes debugging and diagnosis easier, but actually speeds up implementation time. With data not locked in a proprietary or binary data format, developers can easily check to see that their processes are producing the correct results.

Easily internationalised

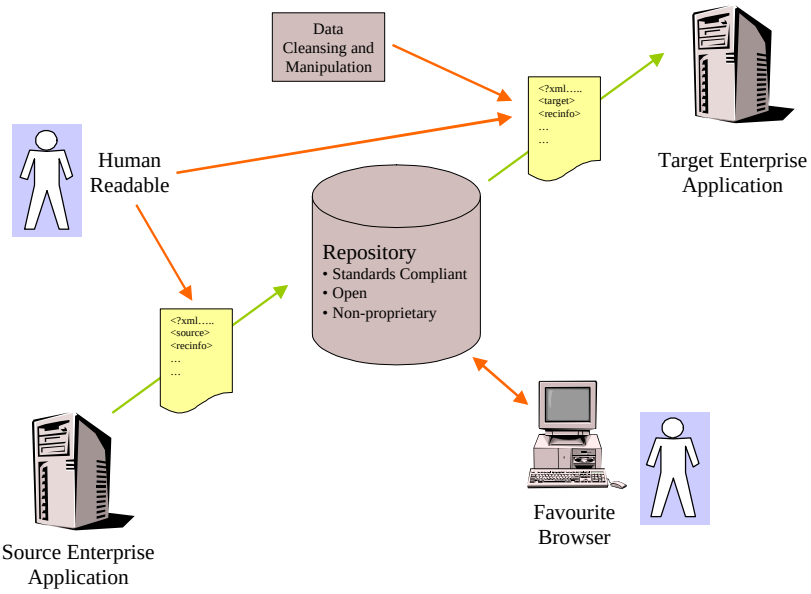
One of the drawbacks to many other formats is that they don’t easily support the needs for internationalisation and localisation. Currently, in other languages, it is difficult to represent information contained in a Unicode alphabet. XML, as part of its initial specification, inherently supports these needs. XML syntax allows for international characters that follow the Unicode standard to be included as content in any XML element. These can then be marked up and included in any XML-based exchange.

While XML gives the freedom to use any Unicode character within the text of an XML document (as content for elements), it strictly limits the characters that can be used as names for XML elements.

Using an XML Repository for Data Migration

The repository contains all the source system data, and by using XML the structure of the data is also preserved. As such, at any time in the future, the data owners will be able to use the repository to supply any incomplete or missing information. The repository therefore provides the longevity to the source data that the business requires. By storing it in a neutral format continued ease of access to it via standard industry browsers is provided.

The repository also provides the platform and source data for the ongoing migration to the target application, facilitating any data cleansing requirements prior to loading the data. This then provides a highly efficient mechanism to enable the population of the target application with the correct data and within a timeframe which the business allows.



Key Benefits

Use of native application tools to reduce cost and risk

The use of native application tools to export and import data to and from the XML repository improves data quality and reduces risk and costs.

Data cleansing

Data Cleansing is supported by combining the source system and target system Schemas with an industry-standard XML schema checker to check the validity and integrity of the XML objects prior to loading them into the target system.

Data cleansing can take a number of forms, such as:

- Renaming of objects
- Modification of the unique identifiers of the objects to conform to another format
- Removal of objects that are not referenced by any other objects
- Insertion of default values into fields within the object
- Modification of the data held within each object to facilitate its conversion to the target format

This data cleansing is facilitated by the fact that the object is in XML. Being highly structured, it is easy to obtain the contents of an element for further processing.

Supports big-bang and incremental data loads

The data is extracted from the source system and stored in the repository as business objects. Therefore, either all or a subset of the objects can be processed at any point in time. The data load framework could subsequently track the Business Objects that have been loaded into target system, hence supporting both incremental and big bang data migration.

Source system data archived in standards-based neutral format

Existing users will be able to access their source system data even when the source system has been retired. Therefore, the business can accept the new system even though some of the source system data has not been migrated to the target system, hence speeding up the analysis and test phases.

Reduced time to perform big-bang data migrations

A full data migration process requires:

- Object export from Source system
- XML manipulation
- Object import into Target

The use of the repository as part of the above process reduces the elapsed time needed to complete a big-bang data migration.

Avoid repetitive source system exports

Data loads often require multiple passes of the data to address the rules embedded in the applications within the target system. Using the repository removes the need to perform multiple repetitive object exports.

Additional incremental loads do not need the source system

Once the main data load has been completed, additional data can be loaded. An incremental load can be performed without the need to access the source system application. This allows the source system to be retired, while access to the data is maintained, in readiness for the target application to match the functional requirements of the business.

Secure intellectual capital

Any organisation will be rightly concerned about turning off the source system and losing the intellectual capital stored within it. The repository, combined with the application rules, ensures that the intellectual capital of the organisation is secured. All object data contained within source system will be available as part of the XML version of the object. This means that any reports generated by the source system could be replicated from information contained within the repository, unless such reports rely on data external to the original source system objects.

Using XML leads to a successful data migration

Using XML provides a fresh project approach that significantly reduces the risk, cost and elapsed deployment time of data migration. XML also delivers a number of other key benefits over and above the traditional ‘point to point’ migration. By utilising technologies that will provide ongoing benefit to the organisation after the migration activity has been completed, the necessary cost of data migration is turned into an investment for the future.

Tony Holtham is an independent Consultant and a member of the Solass Associates Program. He can be contacted through Solass on +44 870 744 8765 or by email to tony.holtham@solass.com